The p-Adaptive Biem Approach for Two-Dimensional Elasticity Analysis

M. CERROLAZA and E. ALARCON
MC. Universidad Central de Venezuela.—EA. Universidad Politecnica de Madrid

This paper presents the development and application of the p-adaptive BIEM version in elastostatics. The basic concepts underlying the p-adaptive technique are summarized and discussed. Some Pascal pseudocodes which show the way how such a technique can be implemented easily in microcomputers are also provided. Both the applicability and the accuracy of the method proposed here are illustrated through a numerical example.

INTRODUCTION

The classical Boundary Integral Equation Method (BIEM) is developed generally by using an isoparametric interpolation criteria. However, it has been shown recently [1, 2, 7, 8, 10, 17, 20] that there also exists an interesting alternative approach by using the so-called *"p-adaptive"* way of analysis. Some other authors [20, 22, 23] have also worked in developing the h-adaptive BIEM version applied to two-dimensional elasticity.

Thus, the advantages of the boundary methods and the spectacular outcomes obtained in the applications of the self-adaptive FEM techniques [3–5, 14, 18, 24, 26] merged together have shown to be very attractive in the solution of both potential and elastostatics problems. The reader interested in a survey of self-adaptive FEM is referred to [5].

The idea is to reduce the numerical effort to a minimum by controlling the degree of accuracy of the solution. To do that it is necessary, first of all, to have a measure of that degree when the problem have just been solved, i.e., an *a-posteriori* error estimate. Moreover, one needs to know *"where"* the solution fails to meet the prescribed degree of accuracy in order to be able to refine those boundary regions mostly involved in such a degree of accuracy. A mathematical criterion, *the local indicator*, has proven to be useful to achieve this goal.

SUMMARY OF CLASSICAL BIEM FORMULAE

As it is well-known, the so-called "direct approach" in the BIEM may be stated through the Somigliana integral identity [6], which is written as (ignoring body forces)

$$\mathbf{C}(P)\cdot\mathbf{U}(P) + \int_\Gamma \mathbf{T}^*(P, Q)\cdot\mathbf{u}(Q)\cdot d\Gamma(Q) = \int_\Gamma \mathbf{U}^*(P, Q)\cdot\mathbf{t}(Q)\cdot d\Gamma(Q). \quad (1)$$

The above expression shows the reciprocity relationship between the present state of traction $\mathbf{t}(Q)$ and displacement $\mathbf{u}(Q)$ at a point $Q \in \Gamma$ and a fundamental solution defined by a unitary concentrated load tensor applied at a point $P \in \Gamma$, whose mechanical state is reflected by both the tractions $\mathbf{T}^*(P, Q)$ and the displacements $\mathbf{U}^*(P, Q)$. $\mathbf{C}(P)$ is a matrix related to the local geometric properties of the surface around the source point P [15].

In the two-dimensional elastostatics case, the expressions for $\mathbf{U}^*(P, Q)$ and $\mathbf{T}^*(P, Q)$ are

$$\mathbf{U}^*(P, Q) = \frac{1}{8\pi G(1-\nu)}[(3-4\nu)\log r^{-1}\delta_{ij} + r_{,i}r_{,j}], \quad (2)$$

$$\mathbf{T}^*(P, Q) = -\frac{1}{4\pi(1-\nu)}\frac{1}{r}[[(1-2\nu)\delta_{ij} + r_{,i}r_{,j}]r_{,n} - (1-2\nu)(r_{,i}n_j - r_{,j}n_i)], \quad (3)$$

where $r_{,i}$ expresses the partial derivative of r with respect to the ith direction, n_i is the vector normal to the surface, and r is the distance between the source point P and the dummy point Q. If the functions $\mathbf{T}^*(P, Q)$ and $\mathbf{U}^*(P, Q)$ are interpreted in the sense of weighting functions, then

it should be possible to interpolate $\mathbf{u}(Q)$ and $\mathbf{t}(Q)$ via the classical approach of well-known projective methods. The state of tractions and displacements over the boundary Γ is represented by

$$\begin{aligned} \mathbf{u}(Q) &= \bar{\mathbf{u}}(Q), \qquad Q \in \Gamma_u, \\ \mathbf{t}(Q) &= \bar{\mathbf{t}}(Q), \qquad Q \in \Gamma_t, \\ \mathbf{u}(Q) &= \ ?\ , \qquad Q \in \Gamma_t, \\ \mathbf{t}(Q) &= \ ?\ , \qquad Q \in \Gamma_u, \end{aligned} \tag{4}$$

where Γ_u and Γ_t represents the kinematic and static boundary, respectively; $\bar{\mathbf{u}}(Q)$ and $\bar{\mathbf{t}}(Q)$ are the known fields of variables over the boundary. Thus, the discretization of the boundary Γ into elements and the unknown fields interpolation gives the well-known matrix relation

$$\mathbf{A} \cdot \mathbf{X} = \mathbf{F}, \tag{5}$$

where the boundary conditions of Eq. (4) have already been imposed. $\mathbf{X}$ collects the boundary unknowns and $\mathbf{A}$ and $\mathbf{F}$ are computed through numerical integration of the influence coefficients. The reader interested in the details of the formulae may see, for instance, references [6, 12, 13, 25].

The classical BIEM version is based usually upon isoparametric interpolation criteria together with enough fine-boundary discretization, which one expects it would be able to catch the local state of the unknown fields. A unique set of points (nodes) is defined over the boundary Γ in order to be used for: a) to define the piecewise discretization; b) to support the local interpolation functions; c) to define the boundary conditions; and d) as a set of source points at which the fundamental solution is placed in order to generate the system of integral equations of Eq. (5).

In the following sections, the p-adaptive BIEM version will be proposed and discussed.

p-ADAPTIVE BIEM

P-HIERARCHICAL INTERPOLATION FUNCTIONS

Once the geometry and the boundary conditions have been established, the first step in a self-adaptive numerical method is the proper choice of a p-hierarchical family of interpolation functions [18, 24] (h-hierarchical in the h-adaptive case [22, 23]).

As it is known, one of the main advantages of the hierarchical families is that the introduction of a new interpolation function $H^m(Q)$ does not modify the influence coefficients of the system of equations generated with functions $H^n(Q)$ of lower order than $H^m(Q)$, where $n = 1, \ldots, m - 1$. In this work, the Legendre hierarchical family [7, 8] has been used, since it produces the best matrix conditioning.

The expression for the Legendre family is as follows:

$$H^0(\xi) = \frac{1}{2}(1 - \xi), \tag{6}$$

$$H^1(\xi) = \frac{1}{2}(1 + \xi),$$

$$H^n(\xi) = \frac{1}{(n-1)!}\frac{1}{2^{n-2}}\frac{d^{n-2}}{d\xi^{n-2}}[(1 - \xi^2)^{n-1}], \tag{7}$$

$$n > 1, \qquad \xi \in [-1, 1].$$

On the other hand, the introduction of new interpolation functions given in Eq. (7) presupposes the choice of new source points, in order to place the fundamental solution at them. This set of new points should be selected in such a way as to satisfy, at least, the following two recommendations [10]:

1. They should be as far apart as possible from previous source points in order to avoid ill-conditioning of the influence matrix.
2. The new equation must reinforce the corresponding dominant element within the influence matrix.

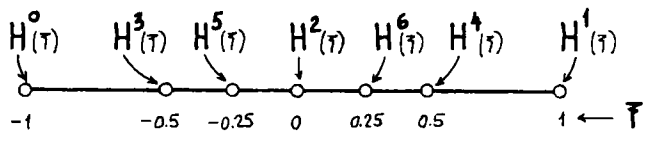
Thus, by following this criteria, the linear interpolation functions of Eq. (6) are placed at the outermost points of the boundary elements, as shown in Figure 1. Higher-order functions (see Eq. (7)) are placed at the middle of the segment, the previous fourth, the later fourth, and so on.

THE SYSTEM OF EQUATIONS: NEW p-ADAPTIVE BLOCKS

When dealing with a p-adaptive method, the idea can be expressed as follows: *a certain number of new interpolation functions should be introduced in such a way as to obtain the maximum accuracy of the numerical solution by employing the minimum number of degrees of freedom involved in the present refinement step.* The idea stated above is accomplished easily with the helpful assistance of some error measures, called "indicators" and "estimator," which will be included and discussed further [8, 11].

In each refinement step, the analysis procedure will produce new influence matrices by "bordering" the previous ones with new p-adaptive influence coefficients. Since the new interpolation functions, as previously discussed, do not perturb the already existing coefficients, the previous influence matrix remain totally valid and, as a consequence, no previous work is loss.

FIGURE 1. p-Adaptive source points for two-dimensional hierarchical interpolation functions.



In order to illustrate the formation of the p-adaptive system, consider that the equations of a certain iteration "k" are expressed by

$$\mathbf{Au} = \mathbf{Bt}, \tag{8}$$

where the elements A_{ij} and B_{ij} are written as follows:

$$\begin{aligned} A_{ij} &= \int_{S_e} T_{ij}^*(P, Q) H^j(Q)\, ds(Q), \\ B_{ij} &= \int_{S_e} U_{ij}^*(P, Q) H^j(Q)\, ds(Q), \end{aligned} \tag{9}$$

in which, for the sake of clarity, the term $C_{ij}(P)$ has been omitted. In expression (9), the term $H^j(Q)$ is the last interpolation function included in refining element S_e, and $T_{ij}^*(P, Q)$ and $U_{ij}^*(P, Q)$ are the fundamental solutions (described in the beginning) defined at a source point P which supports equation i.

Thus, in order to refine the system of equations in the iteration "k" (see Eq. (8)), it becomes necessary to calculate three new boxes of integral coefficients (see Eq. (11)):

Box a: influence of new interpolation functions $H^m(Q)$ $(m > j)$ when they are integrated from already existing source points.

Box b: influence of already existing interpolation functions $H^m(Q)$ $(m \leq j)$ when they are integrated from new source points.

Box c: influence of new interpolation functions $H^m(Q)$ $(m > j)$ when they are integrated from new source points.

So, the new system of equations will look as

$$\mathbf{A^*u^*} = \mathbf{B^*t^*} \tag{10}$$

or, more graphically [1]:

$$\mathbf{A^*}\ (\mathbf{B^*}) = \begin{array}{|c|c|c|} \hline \mathbf{A}\ (\mathbf{B}) & \text{Box a: New functions from old source points} & \text{already existing source points} \\ \hline \text{Box b: old functions from new source points} & \text{Box c: New functions from new source points} & \text{new source points} \\ \hline \text{already existing functions} & \text{new functions} & \end{array} \tag{11}$$

Once the system given by Eq. (10) has been reordered, according to the boundary conditions, one obtains

$$\left[\begin{array}{c|c} \mathbf{K}_{i,i} & \mathbf{K}_{i,i+1} \\ \hline \mathbf{K}_{i+1,i} & \mathbf{K}_{i+1,i+1} \end{array}\right] \left[\begin{array}{c} \mathbf{X}_i \\ \hline \mathbf{X}_{i+1} \end{array}\right] = \left[\begin{array}{c} \mathbf{F}_i \\ \hline \mathbf{F}_{i+1} \end{array}\right], \tag{12}$$

where i now stands for iteration. Note that from the system of equations (8) of iteration i, we have obtained a new system (12) for the present iteration $i + 1$, which includes the previous system. Unfortunately, since the weighting functions T_{ij}^* and U_{ij}^* are defined globally, it is easy to verify that

$$\mathbf{K}_{i,i+1} \neq [\mathbf{K}_{i+1,i}]^T \tag{13}$$

which, obviously, forces the calculation of the three boxes a, b, and c already mentioned.

With the aim to facilitate the understanding of the p-adaptive procedure proposed, some pseudocodes are included here. Even though a pseudocode is usually independent of the programing language, we prefer to use the PASCAL language, since it is quite suited in describing both computer tasks and procedures.

The pseudocode illustrated in Figure 2 describes the basic structure of the main code, called *Biem-p-adap*. The second code (see Figure 3) contains the general tasks which must be performed in order to obtain a p-adaptive system of equations. The bracketed notes are comments to the code, while the language "reserved words" are typed boldface. The tasks that the code must execute are typed in "italic." These tasks can be actions executed by procedures, Boolean functions, etc., as well as local sets of sentences.

ASSEMBLING THE SYSTEM OF EQUATIONS

The *Biem-p-adap* code produces an internal system of nodal codification, by assigning each node an alphanumeric code as a function of its geometry (smooth/corner boundary) as well as the boundary conditions of the elements meeting such a node.

Figure 4 shows some of the element types available in the program.

Once the system of nodal codification has been generated, the assembly of the system of equations is guided easily by such alphanumeric codes. The system of equations is generated by assembling the contributions of a boundary element each time it is "viewed" from a determined source point.

The PASCAL pseudocode in Figure 5 shows the procedure followed by the program for the element assembly. Only "ST" element is included.

ERROR EVALUATION

The final formulae developed by the authors for both the evaluation of the numerical error and how to guide the p-adaptive process is included herein. The reader interested in further details may see, for instance, references [8, 9, 10, 19].

The local error indicator which governs essentially the refinement process may be written as

$$\| e \|_{kj}^2 = \frac{\left[\int_{\Gamma_k} H_j^{n+1}(Q)\, r_j(Q)\, d\Gamma(Q)\right]^2}{\int_{\Gamma_k} H_j^{n+1}(Q)\, L_j(P, Q)\, H_j^{n+1}(Q)\, d\Gamma(Q)}, \tag{14}$$

```
Program  Biem_p_adap ;

{   Main Program :
    Analysis of two-dimensional elasticity problems by the
    p-adaptive BIEM version.
    Developed by:      CERROLAZA M.
                       ALARCON   E.                    }

Const
    Setup some constant values ;

Var
    Definition of data structure ;

Begin  { start of main program }

    Initialize data structure ;
    Generate numerical integration tables ;
    Input all the geometric and loading data ;
    Generate the set of internal nodal codes ;

{    perform the linear interpolation step    }

    Generate and assemble the system of equations for
       the linear step ;
    Save the already generated system on permanent storage ;

    repeat
       Display the post-menu ;

{    the user must select an "option" at the displayed menu    }

       case  option  of

       print_display :
          begin
             Decodificate the solution ;
             Print/display the solution ;
          end;

       graphics :
          begin
             Decodificate the solution ;
             Display a graphic interpretation of the solution ;
          end;

       displacements :
          Compute displacements at selected internal points ;

       stresses :
          Compute stresses at selected internal points ;

       p_adaptive_cycle :
          begin
             Compute and display the global estimator ;
                { see ec.  16 }

{    the user must decide if the refinement either continues
     or not.....}

             if proceed_with_further_refinement then
             begin

                step:= step + 1 ;
                Compute local indicators ;  { see ec. 14 }
                Introduce new source points at elements marked
                   by indicators;
```

FIGURE 2. Pseudocode for main program *Biem_p_adap*.

```
                Read the system coefficientes of the previous
                   iterations from permanent storage ;
                Compute and assemble new p_adaptive
                   coefficients for the present iteration ;
                Save the already generated system on permanent
                   storage ;
                Obtain the p_adaptive solution by solving
                   the system of equations ;
             end;
          end;  { p_adaptive cycle .... }

       save :
          Save the present state of data structure on
             permanent storage ;

       exit :
          Exit the program ;

       end;      { case option of .......}
    until option = exit ;
End.
```

FIGURE 2. Continued

with

k = boundary element which is being refined in the present step;
j = unknown variable which is being refined on the element k;
r_j = residual function obtained on element k;
H_j^{n+1} = new interpolation function to be included in the present step; and
L_i = vectorial integral operator based on Somigiliana's identity.

Expression (14) allows us to determine what element and which variables need to be refined in order to get the maximum accuracy with the minimum number of degrees of freedom in the present step.

The residual function $\mathbf{r}$ could be written as

$$\mathbf{r}(P) = \mathbf{C}(P)[\hat{\mathbf{u}}(P) - \mathbf{u}^{\text{comp}}(P)], \tag{15}$$

where $\hat{\mathbf{u}}(P)$ is the numerical solution obtained by interpolation of the actual variables inside the elements which are going to be refined. Also, $\mathbf{u}^{\text{comp}}(P)$ are the computed values obtained when the fundamental solution is placed in the same points.

The global error estimator is calculated by means of the H_0 Hilbert norm [20, 21], which has the following form:

$$\| E \|_0 = \left[\int_\Gamma \sum_{i=1}^{N_d} r_i^2(P)\, d\Gamma(P) \right]^{1/2} \tag{16}$$

where the integral in Eq. (16) is now extended over the whole boundary, and N_d is the dimensionality of the problem considered (N_d = 1, 2, 3 in potential, two-dimensional, and three-dimensional elastostatics problems, respectively).

USER-PROGRAM COMMUNICATION

The working procedure proposed herein, as it will be shown below, is of a sequential nature performing hierarchical analysis and stopping the process as soon as the desired accuracy has been reached.

Some initial developments [3, 16, 19] in finite elements were oriented towards the implementation of automatic codes running on large computers. However, it is clear that an interactive procedure is more suited to engineering analysis of practical applications and has the advantage that it can be implemented on simple microcomputers. In what follows, we shall try to show how the advantages in having a personal computer, a p-adaptive processor, and a powerful numerical technique such as the BIEM, can be collected together to produce a "user-friendly" and efficient computer code.

One of the main attraction of personal computers is, undoubtedly, their user friendliness. The computer code developed during the present research was designed with this goal in mind, in order to facilitate the user-machine interaction.

The code structure is completely interactive and, therefore, the whole sequence of actions is guided by the user's choice of options from some menus, which are structured according to a hierarchical organization or, in other words, each menu is able to call others [8, 10]. First, the user is requested to select one option at the Main menu displayed in Figure 6. Of course, if all the relevant data was defined previously, it is possible either to proceed with the analysis process itself—by calling the processor module—or to plot the boundary information. If the user wants to input/modify/display any set of input data, the program will display a second-

```
Procedure  Adaptive (List of variables);

Var
    Definition of local variables ;

Begin
    for sub := 1 to number_of_subregions do begin
        Recover the set of boundary elements which define the
            present subregion ;

        for ne := 1 to number_of_elements_of_sub  do begin
            Recover the geometry and boundary conditions of
                the present boundary element ;
            Determine if there exists any new function on
                element ne ;
            if new_function_is_defined  then begin

{  compute and assemble coeff. of block (a), see eq. 11  }

                for sp:= 1 to source_points_in_previous_steps do
                begin
                    if sp_is_contained_in_element_ne then
                        Compute A_ij and B_ij  with the bi-cubic
                            transformation quadrature
                            (see refs. [ 9,10 ])
                    else
                        Compute A_ij and B_ij with special
                            quadratures ;

                    Assemble coefficients into influence matrix
                        and load vector ;
                end;
            end;   { if new_function_is_defined .... }

{   compute and assemble coefficients of blocks (b) and (c),
    see eq. 11     }

            for sp := source_points_in_previous_steps + 1  to
                      actual_number_of_source_points do begin

                if sp_is_contained_in_element_ne then
                    Compute A_ij and B_ij  with the bi-cubic
                        transformation quadrature

                else
                    Compute A_ij and B_ij with special
                        quadratures ;

                Assemble coefficients into influence matrix
                        and load vector ;
            end;
        end;   { number_of_elements_of_sub ..... }

        Generate and assemble diagonal coefficients for new
            interpolation functions ;
    end;  {   number_of_subregions ...}
End;      {   procedure adaptive       }
```

FIGURE 3. Pseudocode for the generation of the p-adaptive system of equations.

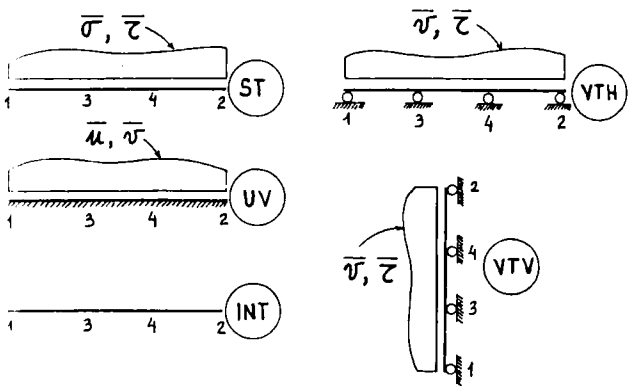


FIGURE 4. Some element types available for two-dimensional elasticity analysis.

level menu, which allows for easily manipulation of data sets resident on permanent storage.

The graphics facilities of the program are provided in the Graphics menu (Figure 7), which allows a considerable number of user-controlled actions in two-dimensional and three-dimensional models. It is of the utmost importance when dealing with discretization, especially in three-dimensional models. Note that this menu is a sublevel one and, therefore, the user can go back to previous menus, correct data (if desired), and display the model again.

The program is based on the p-adaptive BIEM version and, in view of this, it is important that the user be "informed" of the refinement process. For this reason, the program was designed to allow user's control of the pro-

FIGURE 5. Pseudocode for element assembly into the system of equations.

```
Procedure  Assemblage (List of variables);

{  this procedure includes only the "ST" element type,
   see figure 4  }

Var
    Definition of local variables ;

Begin

{  assemble terms due to boundary conditions   }

    if source_point_is_a_new_source_point then
        for node := 1 to max_number_of_geometric_nodes_in_
                                                  element  do
            if node_exists then
                for function := 1 to max_number_of_degrees_of_
                                freedom_per_source_point do

                    if function_is_being_refined then
                        Assemble coeff. B_ij into the load vector;

{  assemble terms due to unknown fields interpolation   }

    for sp := 1 to all_source_points_in_element do
    begin
        Recover nodal code of source point sp ;

        case nodal_code_of_sp  of

        known :  { all displacements are known at source point }
            Assemble coeff. A_ij into the load vector ;

        unknown : { all displac. are unknown at source point }
            Assemble coeff. A_ij into the influence matrix ;

        mixed : { at least one displacement is known at
                  source point  }
            Assemble some A_ij into the influence matrix and
                the remaining into the load vector ;

        end;
    end;   { number_of_source_points_in_element   }
End;       { procedure assemblage                 }
```

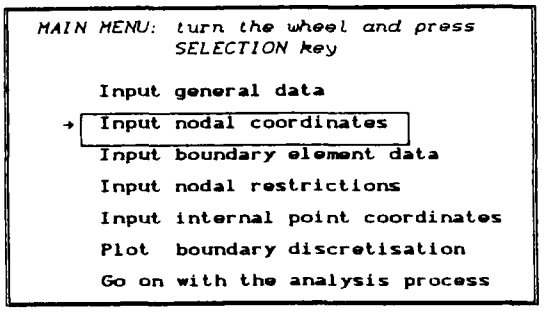


FIGURE 6. Main menu: preprocessor options.

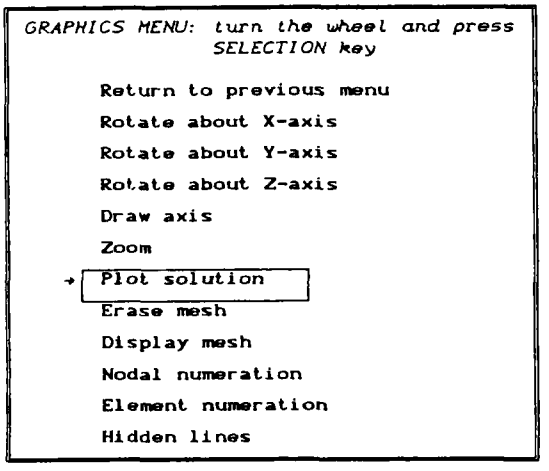


FIGURE 7. Graphics menu for graphics preprocessor.

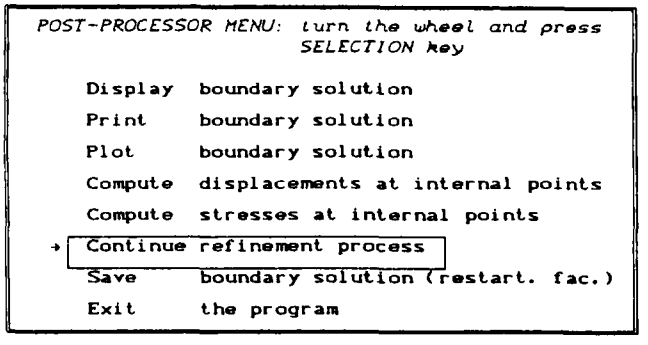


FIGURE 8. Postprocessor menu.

cess, i.e., at each refinement step the program displays a Postprocessor menu, as shown in Figure 8. This menu is also interactive in order to allow the execution of different actions without any priority level between them. The option labeled "*continue refinement process*" includes both the global estimator and local indicators calculation.

NUMERICAL RESULTS

In order to illustrate the power and flexibility of the p-adaptive BIEM version, the numerical analysis of an L-shaped plate subjected to boundary tractions is presented below. This case example is interesting due to the fact that the fields of stresses show a tendency towards *infinite values at the re-entrant corner identified* with the symbol A. This example case was analyzed with both a 120 boundary-element mesh (20 elements/side) and the p-refinement approach proposed here.

Figure 9 shows both the geometry and boundary conditions of the plate, which has been discretized with only six macro-boundary elements whose numbers are circled. This will be the initial mesh.

The global convergence rates of the L-plate are shown in Figure 10. An h-refinement, which gives a convergence rate of 0.70, was performed over the initial mesh in order to be compared with the p-refinement ones. The p-adaptive convergence rate (line with circles) was 2.0, i.e., 2.87 times faster than the h-refinement one, even in the presence of singularities. The p-complete approach (introduction of all interpolation functions in each refinement step) gave a convergence rate of 1.60 times faster than the h-refinement one, but it was slower than the p-adaptive one. This facts shows that an effective selective criteria must be used in order to decide which new interpolation functions must be included to improve the accuracy of the numerical solution. Also note that the geometric singularities (those which do not involve sudden changes in boundary conditions) can be treated effectively with the p-adaptive approach. Observe that the singular point A is bounded by only two macro-elements, it not being necessary to define addi-

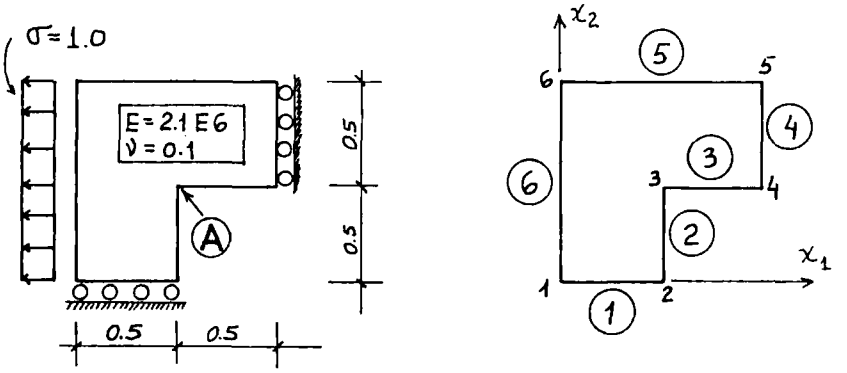


FIGURE 9. (a) Geometry and boundary conditions. (b) p-Adaptive discretization.

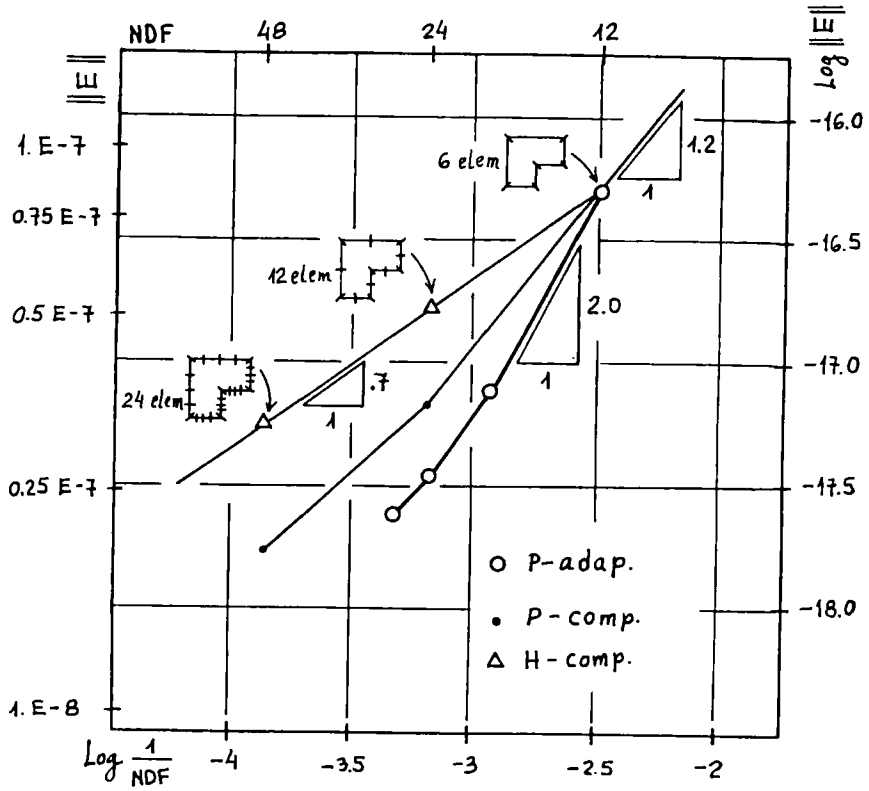
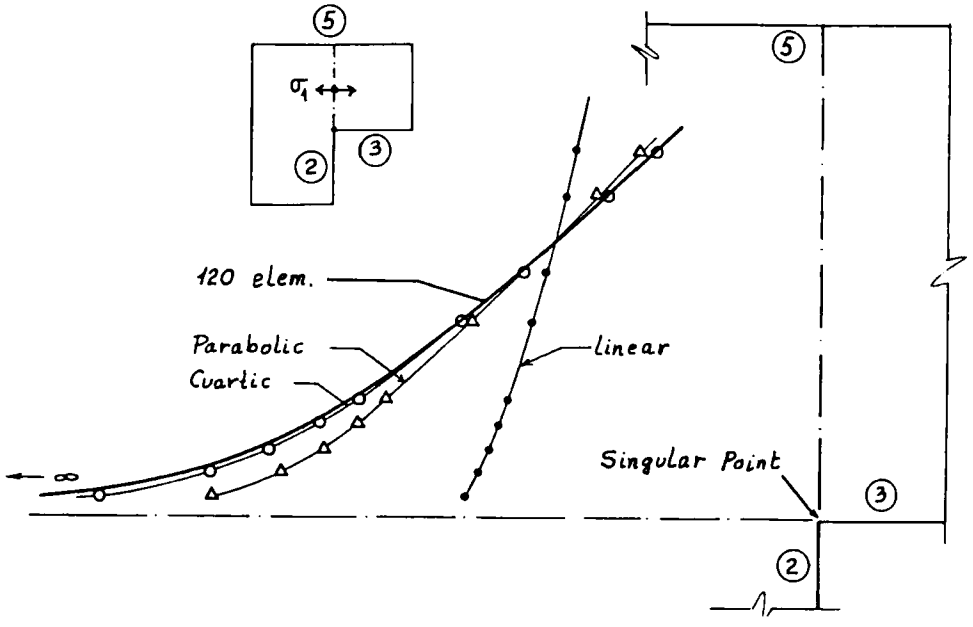


FIGURE 10. Comparison of global rates of convergence: *p* refinement *vs h* refinement.

tional small elements in order to catch the stress concentration.

Figure 11 collects the evolution of the internal stresses calculated along the vertical line $x_1 = 0.5$. The thick line represents the 120-boundary-element solution while the lines with symbols show the stresses obtained in some p-refinement steps. Note that the stresses at or near the singular point A were quite well catched when they were compared with the ones provided by the 120-boundary-element solution.

FIGURE 11. Stresses σ_1 on $x_1 = 0.5$.



CONCLUSIONS

The analysis of some numerical examples have shown that the p-adaptive BIEM version proposed here is powerful and well-suited when dealing with singularities. The indicators and estimator used in controlling the refinement process have shown to be very useful when increasing the space of polynomial interpolation functions over the initial mesh. They also produce rates of convergence faster than those obtained with the h-adaptive approach.

The computer codes developed during this research are *"user-friendly"* and they allow the analyst to control the refinement process more effectively. The pseudocodes included contain the general steps in developing a p-adaptive BIEM processor.

Part of this work has been done under the support of the Cosejo de Desarrollo Cientifico y Humanistico (CDCH, Venezuela) and the Instituto de Cooperacion Iberoamericana (ICI, España).

REFERENCES

1. Alarcón, E. and Reverter, A., p-adaptive boundary elements. *International Journal of Numerical Methods in Engineering* 23:801–829 (1986).
2. Alarcón, E., Reverter, A., and Molina, J., Hierarchical boundary elements. *Computers and Structures* 20(1–3):151–156 (1986).
3. Babuska, I. and Rheinboldt, W.C., Error estimates for adaptive finite-element computations. *Siam Journal of Numerical Analysis* 15(4):736–754 (1978).
4. Babuska, I. and Miller, A., A-posteriori error estimates and adaptive techniques for the finite-element method. Technical Note Bn-968, Institute for Physical Science and Technology, University of Maryland, MD, 1981.
5. Babuska, I., Zienkiewicz, O.C., and Gago, J., Accuracy Estimates and Adaptive Refinements in Finite-Element Computations. John Wiley & Sons, NY, 1986.
6. Brebbia, C.A., Telles, J.C., and Wrobel, L.C., *Boundary-Element Techniques: Theory and Applications in Engineering*. Springer-Verlag, Berlin, 1984.
7. Cerrolaza, M. and Alarcón, E., p-adaptive boundary elements for three-dimensional potential problems. *Communications on Applied Numerical Methods* 3:335–345 (1987).
8. Cerrolaza, M. and Alarcón, E., Elastostatics p-adaptive BE for micros. *Software for Engineering Workstations* 4(1):18–26 (1988).
9. Cerrolaza, M. and Alarcón, E., A bi-cubic coordinate transformation for the evaluation of Cauchy principal-value integrals. *International Journal of Numerical Methods in Engineering* (accepted for publication, 1988).
10. Cerrolaza, M., P-adaptive Boundary Elements: development and applications in Potential Theory and Elastostatics. Doctoral Dissertation (in Spanish), Polytechnical University of Madrid, Madrid, Spain, 1988
11. Cerrolaza, M. and Alarcón, E., Further applications of p-adaptive BIEM. In *IX International Conference of BIEM*, Brebbia, C.A. and Wendland, A. (eds.), Springer-Verlag, Stuttgart, 1988.
12. Cruse, T.A., An improved boundary integral equation method for three-dimensional elastic stress analysis. *Computers and Structures* 4:741–754 (1974).
13. Cruse, T.A., Mathematical foundations of the BIEM in Solid Mechanics Report AFOSR-TR-77-1002, Pratt & Whitney Aircraft Corporation, 1977.
14. Gago, J.P., A posteriori error analysis and adaptivity for the finite-element method. Doctoral Thesis, University of Wales, Swansea, 1982.
15. Hartmann, F., Computing the C-matrix in non-smooth boundary points. In *New Developments in Boundary-Element Methods*, Brebbia, C.A. (ed.), pp. 367–379. Butterworths, London, 1983.
16. Kelly, D.W., Gago, J.P., Zienkiewicz, O.C., and Babuska, I., A posteriori error analysis and adaptive process in the finite-element method. *International Journal of Numerical Methods in Engineering* Parts I and II, 19:1593–1619 (1983).
17. Parreira, P., Self-adaptive p-hierarchical BEM in elastostatic. In *Proceedings of the IX International BEM Conference* Brebbia, C.A. and Wendland, A. (eds.), Springer-Verlag, Stuttgart, 1987.
18. Peano, A.G., Hierarchy of conforming finite elements. Doctoral Dissertation, Washington University, St. Louis, MO, 1975.
19. Peano, A.G., Passini, A., Riccioni, R., and Sardella, L., Adaptive approximation in finite-element structural analysis. *Computers and Structures* 10:332–342 (1979).
20. Rank, E., Adaptive boundary-element methods. In *Proceedings of the IX International BEM Conference* Brebbia, C.A. and Wendland, A. (eds.), Springer-Verlag, Stuttgart, 1987.
21. Reddy, J.N., *An Introduction to the Finite-Element Method*. Mc.Graw-Hill Book, Singapore, 1984.
22. Rencis, J.J. and Mullen, R.L., A self-adaptive mesh refinement technique for boundary-element solution of the Laplace equation. *Computer Methods in Applied Mechanics and Engineering* (1986).
23. Rencis, J.J. and Mullen, R.L., Solution of elasticity problems by a self-adaptive mesh refinement technique for boundary-element computations. *International Journal of Numerical Methods in Engineering* 23:1509–1527 (1986).
24. Szabo, B.A., Basu, P.K., and Rosow, P., Adaptive finite elements based on p-convergence. *NASA Conference Publication* 2059:43–50 (1978).
25. Symm, G.T., Integral equation method in elasticity and potential theory. Doctoral Thesis, London University, UK, 1964.
26. Zienkiewicz, O.C. and Morgan, K., *Finite Elements and Approximation*. John Wiley & Sons, NY, 1983.